CMU-CS-83-158

A Tutorial on Techniques and Applications for Natural Language Processing

Philip J. Hayes and Jaime G. Carbonell

17 October 1983

DEPARTMENT
of
COMPUTER SCIENCE

Carnegie-Mellon University

A Tutorial on Techniques and Applications for Natural Language Processing

Philip J. Hayes and Jaime G. Carbonell

Carnegie-Mellon University

17 October 1983

This research was sponsored in part by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under contract F33615-81-K-1539, and in part by the Air Force Office of Scientific Research under Contract F49620-79-C-0143. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, the Air Force Office of Scientific Research or the US government.

This report is a revised version of a set of notes originally prepared for the Natural Language Tutorial presented in conjunction with the Eighth International Joint Conference on Artificial Intelligence held in Karlsruhe, West Germany in August, 1983.

Table of Contents

List of Figures

1. Introduction

Natural language communication with computers has long been a major goal of Artificial Intelligence both for what it can tell us about intelligence in general and for its practical utility — data bases, software packages, and AI-based expert systems all require flexible interfaces to a growing community of users who are not able or do not wish to communicate with computers in formal, artificial command languages. Whereas many of the fundamental problems of general natural language processing (NLP) by machine remain to be solved, the area has matured in recent years to the point where practical natural language interfaces to software systems can be constructed in many restricted, but nevertheless useful, circumstances. This tutorial is intended to survey the current state of applied natural language processing by presenting computationally effective NLP techniques, by discussing the range of capabilities these techniques provide for NLP systems, and by discussing their current limitations. Following the introduction, this document is divided into two major sections: the first on language recognition strategies at the single sentence level, and the second on language processing issues that arise during interactive dialogues. In both cases, we concentrate on those aspects of the problem appropriate for interactive natural language interfaces, but relate the techniques and systems discussed to more general work on natural language, independent of application domain.

1.1. The nature of natural language processing

We define natural language processing (NLP) to be:

> the formulation and investigation of ***computationally effective*** mechanisms for ***communication*** through ***natural language***.

To take the highlighted phrases in reverse order, first the subject area deals with naturally occurring human languages such as German, French, or English. Secondly, it is concerned with the use of these languages for communication, both communication between people, the purpose for which these languages evolved, and communication between a person and a computer, a convenience which is becoming increasingly feasible and desired, and which is the primary motivation for this tutorial. Thirdly, natural language processing does not study natural language communication in an abstract way, but by devising mechanisms for performing such communication that are computationally effective, i.e. can be turned into computer programs that perform or simulate the communication. It is this third characteristic that sets the natural language processing subarea of Artificial Intelligence, itself a subarea of Computer Science, apart from traditional linguistics and other disciplines that study natural language.

In order to provide sufficient context for readers with backgrounds in other disciplines and to help everyone gain a clearer perspective on the nature of natural language processing (hereinafter NLP), it

is worthwhile to examine the relationship between NLP and two other closely related disciplines: linguistics and cognitive psychology.

Linguistics is traditionally concerned with formal, general, structural models of natural language. Linguists, therefore, have tended to favor formal models which allow them to capture as much as possible the regularities of language and to make the most appropriate linguistic generalizations. Little or no attention was paid in the development of these models to their computational effectiveness. That is, linguistic models characterize the language itself, without regard to the mechanisms that produce it or decipher it. A good example, as we shall see later is Chomskian *transformational grammar* [21, 22], perhaps the best known of all linguistic models, which turns out to be unsuitable as a basis for computationally practical language recognition (although see work by Petrick [53]).

The goal of cognitive psychology on the other hand is not to model the structure of language, but rather to model the use of language, and to do it in a psychologically plausible way, where plausibility here is defined by correlation with experimental results, especially timing studies of language understanding tasks (see Anderson [3] for a good example of the flavor of this approach). This is somewhat closer to the spirit of AI-based NLP in its emphasis on the use of language in communication, but again it is not of primary importance to the cognitive psychologist whether his models are computationally effective. Moreover, the models produced are not often targeted at language understanding per se, but at more general aspects of human cognition and memory organization, with natural language serving only as the vehicle through which these related phenomena are studied.

In addition to relating NLP to the study of language in other disciplines, we should point out a major division that arises within NLP itself. The distinction is between general and applied natural language processing. One can think of general NLP as a way of tackling cognitive psychology from a computer science viewpoint. The goal is to make models of human language use, and also to make them computationally effective. The vehicles for this kind of work are general story understanding, as in the work of Charniak [19], Schank [59], Cullingford [25], Carbonell [9], and others, and dialogue modeling as in the work of Cohen and Perrault [23], Allen [1], Grosz [32], Sidner [66], and others. One of the most important lessons learnt from this work is that general NLP requires a tremendous amount of real-world knowledge, and most of the work just cited is mainly concerned with the representation of such real-world knowledge and its application to the understanding of natural language input. Unfortunately, AI has not yet reached the stage where it can routinely handle the amount of knowledge required for these tasks, with the result that systems constructed in this area

tend to be "pilot" systems which demonstrate the feasibility of a concept or approach, but do not contain a large enough knowledge base to make them work on more than a handful of carefully selected example natural language passages or dialogues.

Applied natural language processing, on the other hand, is not typically concerned with cognitive simulation, but rather with allowing people to communicate with machines through natural language. The emphasis is pragmatic. It is less important in applied NLP whether the machine "understands" its natural language input in a cognitively plausible way, than whether it responds to the input in a way helpful to the user and in accordance with the desires expressed in the input. Typical applications are data-base interfaces as in the work of Hendrix [41], Grosz [34], Kaplan [44], and others, and interfaces to expert systems, as in the work of Brown and Burton [6], and Carbonell (J. R.) [8] and Carbonell (J. G.) et al [13]. Because such systems must operate robustly with real users, in addition to actually processing well formed natural language, they must be concerned with the detection and resolution of errors and misunderstandings by the user.

In this tutorial, we concentrate mainly on applied NLP, since it is representative of what is currently practical in natural language processing. But, as we proceed, a number of points will arise that demonstrate some of the techniques and problems of general NLP.

1.2. The basic problem of natural language processing

If there is one word to describe why NLP is hard, it is ***ambiguity***. It arises in natural language in many different forms including:

- Syntactic (or structural) ambiguity:

  John saw the Grand Canyon flying to New York
  Time flies like an arrow

  Is it John or the Grand Canyon doing the flying? The answer depends on the ambiguous syntactic role of the word "flying" in this example. Again, is time flying, or are we talking about a species of insect called time flies in the second example. It depends whether "flies" is a noun or a verb.[1]

- Word Sense ambiguity:

  The man went to the ***bank*** *to get some cash*
  and jumped in

  Here the word "bank" refers either a repository for money or the side of a river depending on the two different continuations.

- Case

[1] Actually, the second example here has at least six different parsings. See if you can find them all.

*He ran the mile **in** four minutes*
the Olympics

Linguistically, a "case" refers to the relation between a central organizing concept, here an act of running, and a subsidiary concept, here time or location. In both examples, the same preposition, "in", indicates the two quite different relationships.

- Referential

*I took the cake from the table and ate **it**.*

What was eaten, the cake or the table? The answer is "obvious", but, independent of real-world knowledge, "it" could refer to either one.

- Literalness

Can you open the door?
I feel cold.

What are the correct interpretations here? There are some circumstances when the first question might be answered quite reasonably "yes" or "no", e.g. before setting off on a long journey to the place where the door is. On the other hand, it is easy to think of circumstances where the speaker might be very unhappy with such a reply. Again, the second example might be a statement of fact or a request to close a window. The ambiguities here lie in whether to interpret the utterance literally, or whether to treat it as an indirect speech act [64], e.g. an implicit request as in the examples above.

Because of these and other kinds of ambiguity, the central problem in NLP, and this is true for both the general and applied variety, is the translation of the potentially ambiguous natural language input into an unambiguous internal[2] representation, as suggested by Figure 1.

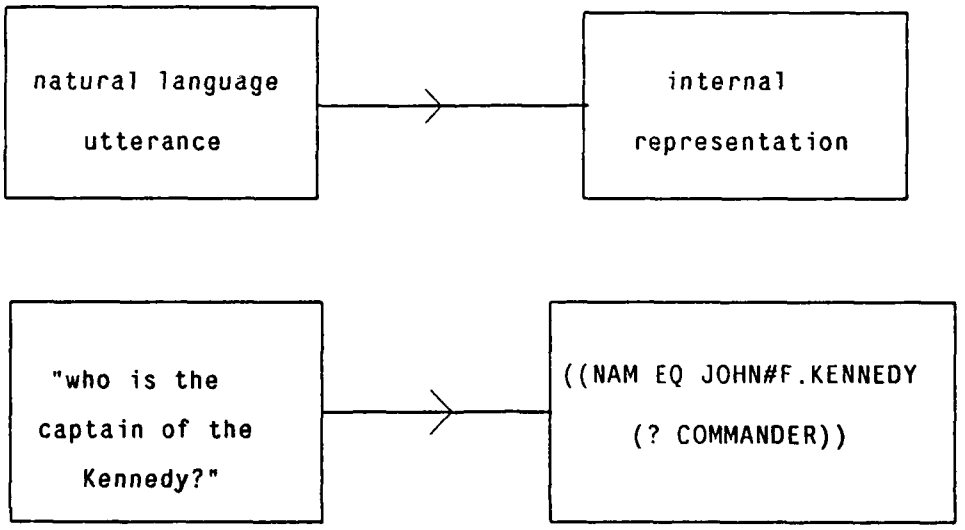

Figure 1: Translation from a natural language utterance into unambiguous internal representation

[2] internal to the program doing the processing, that is.

The second layer of Figure 1 shows an example translation of a natural language database query into an expression in a database query language - the one used by the LADDER [57] system for access to its database of information about U.S. Navy ships. Note how a potentially ambiguous word such as "Kennedy" is resolved into the internal name, JOHN#F.KENNEDY, of a specific ship, or "captain" is resolved into the name, COMMANDER, of a field of the relational database conceptually underlying the LADDER system. The specific internal representation used here is, of course, highly specialized. In general, there is no commonly agreed standard for internal representations, and different types are useful for different purposes, a partial list includes:

- expressions in a database query language (for DB access)
- parse trees with word sense terminal nodes (for machine translation)
- lisp expressions (most often for expert system requests)
- case frame instantiations (for a variety of applications)
- conceptual dependency (for story understanding)

In general NLP, translation of an utterance into an unambiguous internal representation can require inference based a potentially unbounded set of real-world knowledge. Consider, for instance:

Jack took the bread from the supermarket shelf, paid for it, and left.

Coming up with an unambiguous representation for this requires answers to such questions as:

What did Jack pay for? (the referent of "it")
What did Jack leave? (the ellipsed object of "left")

and possibly even:

Did Jack have the bread with him when he left?

To answer these questions, information on supermarkets, buying and selling, and other real-world topics is required. As mentioned above, AI knowledge representation techniques have not yet developed to the stage where they can handle at an acceptable level of efficiency the large quantities of such knowledge required to do a complete job of understanding a large variety of topics. Moreover, even if the knowledge could be represented, unresolved problems in inference techniques remain a barrier to applying the correct knowledge to the input in order to produce the desired unambiguous internal representation. The result is that current general NLP systems are demonstration systems that operate with a very small amount of carefully handcrafted knowledge, specifically designed to enable the processing of a small set of example inputs. The main point of such systems is to investigate the feasibility of certain inference or knowledge representation techniques, rather than to achieve broad coverage in the natural language processing they perform.

Applied NLP systems potentially face exactly the same problem, but they finesse it by taking advantage of certain characteristics of the highly limited domains in which they operate. Suppose the input:

How many terminals are there in the order?

was addressed to an expert system that acted as a computer salesman's assistant. Such a system need not consider many of the potential ambiguities lurking in this example. The word "terminals", for instance, can be assumed to refer to computer terminals, rather than airport terminals or terminal values of a mathematical series. Also, assuming the system processes one sales order at a time, "the order" can be assumed to refer to the current order without considering any others. In general, the technique is to pre-make as many inferences as possible in a way appropriate to the task at hand. For suitable tasks in many restricted domains, this has been used very successfully to reduce the amount of knowledge that must be represented and the number of inferences that must be made to manageable proportions.

By restricting the natural language dealt with by an interface to that required to handle a limited task in a limited domain, it is thus possible to construct performance systems capable of useful natural language communication, and this represents the current state of the art in practical natural language processing. Clearly, this is far from satisfactory, since in particular, each task and domain that are tackled require careful preanalysis so that the required inferences can be pre-encoded in the system, thus making it difficult to transfer successful natural language interfaces from one task to another. Some research (e.g. [34]) is being conducted to improve the portability of current interfaces, but until the problem of pre-encoding inferences is solved in a more general way, the portability issue will be the one that most hinders the widespread use of natural language interfaces.

2. Natural Language Analysis Techniques

In this section, we examine in some detail several of the more common techniques for natural language analysis, i.e. for translating natural language utterances into a unique internal representation. Virtually all natural language analysis systems can be classified into one of the following categories:

- Pattern matching (e.g. ELIZA [72], PARRY [51])
- Syntactically-driven parsing (e.g. ATNs [75])
- Semantic grammars (e.g. LIFER [41], SOPHIE [6])
- Case frame instantiation (e.g. ELI [55])
- Wait and see (e.g. Marcus [48])

- Word expert (e.g. Small [68])
- Connectionist (e.g. Small [69])
- Skimming (e.g. FRUMP [26], IPP [63])

The examples provided with each category are the names of language analysis systems following that approach, or the names of builders of such systems. Of these categories, the first four represent the bulk of the language analysis systems already constructed, and are the only ones we will cover in detail. The reader is encouraged to follow up the references provided for further details of the other methods.

2.1. Pattern Matching

The essence of the pattern matching approach to natural language analysis is to interpret input utterances as a whole, rather than building up their interpretation by combining the structure and meaning of words or other lower-level constituents. The approach is thus *wholistic* rather than *constructive*. With this approach, the interpretations are obtained by matching patterns of words against the input utterance. Associated with each pattern is an interpretation, so that the derived interpretation is the one attached to the pattern that matched. In the simplest case, this arrangement is simply a list of correspondences between equivalence classes of utterances (the ones that match a given pattern) and interpretations (the ones associated with each pattern). In more sophisticated variations of the approach, patterns may involve higher-level constituents or semantic elements, so that some aspects of the interpretation may become constructive, but the basic flavor of the approach still remains to go as directly as possible from the input utterance to the interpretation as suggested by Figure 2.

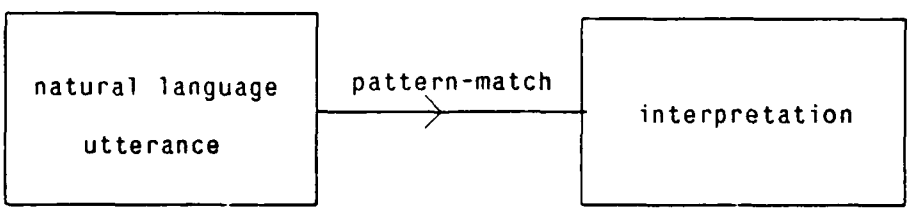

Figure 2: Parsing by pattern matching

One of the simplest and earliest examples of pattern matching parsing was the ELIZA system of Weizenbaum [72]. This system demonstrated how a very simple mechanism in the form of a computer program was able to mimic apparently intelligent behavior. The carefully selected task of ELIZA was

to simulate a Rogerian psychologist as he interviewed a patient. The style of this school of psychology allowed ELIZA to understand virtually nothing of what was said and still, in many cases, come up with convincing replies. ELIZA did not construct an internal representation of its input as such, but instead went directly from the input to its reply, which could therefore be thought of as the "interpretation" of the input. The input was matched by a small set of single-level patterns, each of which was associated with several replies. The first pattern found to match an input led to a selected reply from the associated set; if a pattern matched again on a subsequent input, the next reply was given. Variables in the patterns which would match any sequence of words whatsoever allowed text from the input to be incorporated into the reply. Example patterns and corresponding replies with variables represented by 'X' and 'Y' include:

X always Y	Can you think of a specific example? When? Really, always?
X you Y me	Why do you think I Y you? You like to think I Y you, don't you. Suppose that I did Y you. What would that mean?

These would allow ELIZA (E) conduct the following dialogue with a patient (P):

P: People always stare at me matches first pattern
E: Can you think of a specific example?
P: For instance you keep staring at me matches second pattern with Y matching "keep staring at"
E: Why do you think I keep staring at you
first reply for second pattern with appropriate substitution for Y
P: Because you hate me second pattern again with Y matching "hate"
E: You like to think I hate you, don't you. next (second) reply for second pattern

The simplicity of the matching and reply generation meant that not all conversations with ELIZA went as smoothly as this, but there are several anecdotes about people being fooled into thinking they were talking to a real psychologist.

ELIZA could achieve its results with such a low level of analysis only by ignoring most of what was said to it. To make more complete analyses of the input using the same techniques would require far too many patterns - in the extreme, one pattern for every possible utterance. Moreover, many of these patterns would contain common subelements because they mentioned the same objects or had the same concepts arranged with slightly different syntax. In order to resolve these problems within the pattern matching approach, hierarchical pattern matching methods have been developed in which some patterns match only part of the input and replace that part by some canonical result. Other higher-level patterns can then match on these canonical elements in a similar way, until a top-level pattern is able to match the canonicalized input as a whole according to the standard pattern

matching paradigm. In this way, similar parts of different utterances can be matched by the same patterns and the total number of patterns is much reduced and made more manageable.

The best known example of hierarchical pattern matching is the PARRY system of Colby [24, 51]. Like ELIZA, this program operates in a psychological domain, but models a paranoid patient rather than a psychologist. Using the traditional pattern matching paradigm, PARRY interprets its input utterances as a whole by matching them against a set of about 2000 general patterns. The internal representation into which the input is transformed is a set of updates to a simple model of the paranoid patient's mental state, plus a representation of any factual content of the input. Replies are generated from the updated paranoid model, plus the factual content. However, before the general patterns are applied, PARRY massages its input through a series of eight canonicalizing steps, most of which are based on localized pattern matching. Examples of these steps include:

- canonicalizing rigid idioms ("have it in for" -> "hate")
- noun phrase bracketing using an ATN (see Section 2.2.3)
- canonicalizing flexible idioms ("lend [] a hand" -> "help []")
- clause splitting ("I think you need help" -> "(I think) (you need help)")

Using rules of this form, an input such as:

Do you have it in for me? I want to lend you a hand.

can be canonicalized into a form similar to:

```
(YOU HATE ME) + INTERROGATIVE=?
(I WANT) (I HELP YOU)
```

and an appropriate reply generated by matching against Parry's 2000 general patterns.

As well as matching patterns of words, it is also possible to analyze natural language input by matching patterns of semantic elements, with potentially very powerful results as shown by the machine translation system of Wilks [73]. The goal of this system was to translate English input into French output. To do this it first analyzed its English input into an internal meaning representation from which it could generate the French. This analysis was performed by matching the input against a very general set of patterns such as:

(MAN FORCE MAN)

which matches all events in which a person compels another person to do something. Other general patterns involved people doing things to objects, objects being in certain states, etc.. To allow matches against these patterns, Wilks represented word senses as formulas of the same semantic primitives as appeared in the patterns, so for instance, "interrogate" was:

((MAN SUBJ) ((MAN OBJE) (TELL FORCE)))

i.e. a person forcing another person to tell something, and "crook" was one of the two following possibilities:

((((NOTGOOD ACT) OBJE) DO) (SUBJ MAN))

((((((THIS BEAST) OBJE) FORCE) (SUBJ MAN)) POSS) (LINE THING))

i.e. a person who does bad things, or a long thin thing that a person uses to force animals (normally sheep) to do something. As well as providing an interpretation of the input, the process of matching these formulas against the general patterns also allowed word senses to be disambiguated. So

The policeman interrogated the crook

is analyzed by matching it against the (MAN FORCE MAN) pattern, and this also choose the bad person sense of crook because it matches the second MAN of this pattern. There is also a (MAN FORCE THING) pattern, but this does not match as well because the formula for "interrogate" specifies MAN for its object. Note that the notion of degree of match is present in this system. As we will see later in Section 2.5, this idea makes parsing by pattern matching considerably more powerful, especially when the input contains grammatical errors.

To summarize this section on parsing by pattern matching, the basic paradigm is to recognize input utterances as a whole by matching them against patterns of words, wildcards, and/or semantic primitives. The result of the match is the interpretation of the utterance. Unless a very shallow level of analysis is acceptable, the number of patterns required is too large, even for restricted domains. This problem can be overcome by hierarchical pattern matching in which the input is gradually canonicalized through pattern matching against subphrases. The number of patterns can also be reduced by matching with semantic primitives instead of words.

2.2. Syntactically-Driven Parsing

Syntax deals with the ways that words can fit together to form higher-level units such as phrases, clauses, and sentences. Syntactically-driven parsing is, therefore, naturally constructive, i.e. the interpretations of larger groups of words are built up out of the interpretations of their syntactic constituent words or phrases. In this sense, it is just the opposite of pattern matching in which the emphasis is on interpretation of the input as a whole. The most natural way for syntactically-driven parsing to operate is to construct a complete syntactic analysis of the input utterance first, and only then to construct the internal representation or interpretation. As we will see, this leads to considerable inefficiency, and more recent syntactically-driven approaches have tried to intermix parsing and interpretation.

2.2.1. Parse trees and context-free grammars

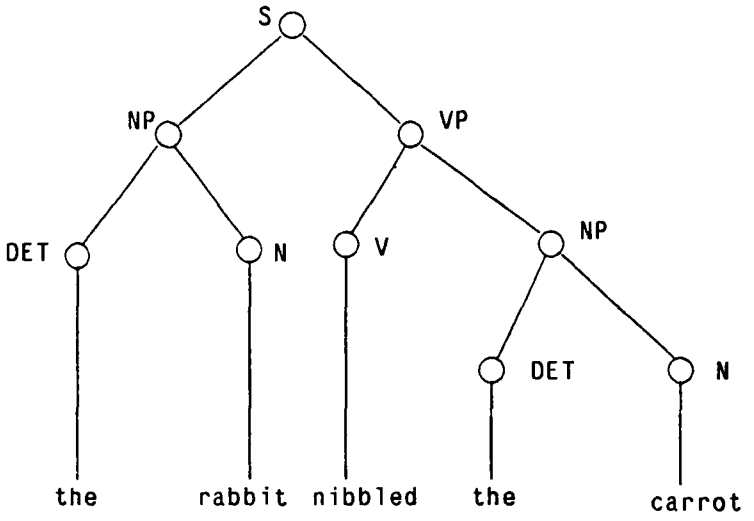

Figure 3: A parse tree for "the rabbit nibbled the carrot"

The most common form of syntactic analysis is known as a *parse tree*. Figure 3 shows a parse tree for the sentence:

The rabbit nibbled the carrot.

The tree shows that the sentence is composed of a noun phrase (subject) and a verb phrase (predicate). The noun phrase consists of a determiner (the) followed by a noun (rabbit), while the verb phrase consists of a verb (nibbled) followed by another noun phrase (the direct object), whose determiner is "the" and whose noun is "carrot".

Syntactic analyses are obtained by application of a *grammar* that determines what sentences are legal in the language being parsed. The method of applying the grammar to the input is called a *parsing mechanism* or *parsing algorithm*. A very simple style of grammar is called a *context-free*[3] grammar, which consists of rewrite rules of the following form:

S -> NP VP
NP -> DET N | DET ADJ N
VP -> V NP
DET -> the
ADJ -> big | green
N -> rabbit | rabbits | carrot
V -> nibbled | nibbled | nibble

As this example shows, context-free grammars have the advantage of being simple to define. They

[3] context-free because the symbol on the left-hand side of a rewrite rule may be replaced by the symbol on the right hand side regardless of the context in which the left-hand side symbol appears.

have been widely used for computer languages, and highly efficient parsing mechanisms [27] have been developed to apply them to their input. However, they also suffer from some severe disadvantages. It should be clear that the above context-free grammar accounts for the parse shown in Figure 3: rewrite rules correspond directly to bifurcations in that tree. While accounting for that and several other good sentences, the grammar also allows several bad ones, such as:

The rabbits nibbles the carrot

The problem here is that the context-free nature of the grammar does not allow agreements such as the one required in English between subject and object. To enforce such an agreement, we would have to have two completely parallel grammars, one for singular sentences and the other for plural. Moreover, a grammar which also allowed passive sentences such as:

The carrot was nibbled by the rabbit

would have to have another completely different set of rules, even though the passive and the active forms of the same sentence have a clear syntactic relation, not to mention semantic equivalence. These duplications are multiplicative rather than additive, leading to exponential growth in the number of the grammar rules. Thus in terms of the number of rules involved, and in terms of being unable to capture related phenomena by related rules, context-free grammars turn out to be quite unsuitable for natural language analysis.[4]

There is one more point to be made with this example, one not specific to context-free grammars, but a serious problem for all syntactically-driven parsing. The above grammar also allows:

The rabbit was nibbled by the carrot.

This is an example of a sentence that is perfectly good syntactically, but makes no sense at all. For utterances that are ambiguous syntactically (and for more comprehensive grammars, syntactic ambiguity is very common), such acceptance of nonsensical interpretations can lead to the highly inefficient generation of multiple parses, only one of which has a reasonable translation into the final internal semantic representation.

2.2.2. Transformational grammar

The problems mentioned above as specific to context-free grammars were tackled by linguists, in particular Chomsky [21, 22] through Transformational Grammar. As shown in Figure 4, their answer was to add another type of rule to a context-free grammar. Space prohibits a full example here, but the basic idea was to use the context free grammar to generate a parse tree just as before, but add

[4] Although recent work by Gazdar [30] and others has shown that these problems of exponential rule growth can substantially be resolved with relatively minor extensions to the context-free formalism, and in particular without going to the transformational framework discussed below, the computational tractability of generalized phrase structure grammar, as the extended formalism is called, has yet to be determined.

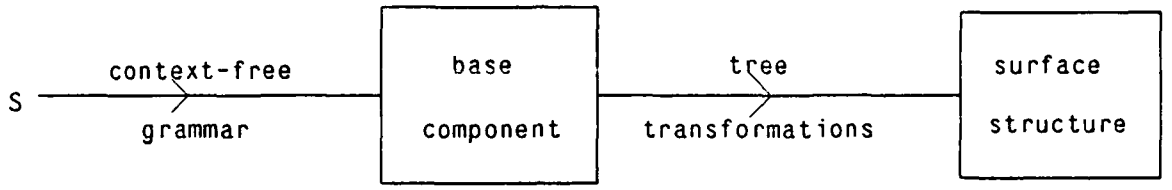

Figure 4: Transformational grammar

onto it certain tags, such as one for a plural sentence. The set of transformations on the parse tree would then rearrange things so that the pluralness was transmitted to all parts of the tree concerned and the required agreements could be enforced. The transformations that enforced agreements were called *obligatory transformations*. A second class of *optional transformations* was used to capture the relations between, for instance, active and passive sentences; the active and passive versions of the same sentence had the same representation in the base component produced by the context-free grammar, but the passive version was the result of applying an extra optional transformation. Transformations are context-sensitive rules that map a parse tree into a related parse tree.

While transformational grammar did a much better job of accounting for the regularities of natural language[5] than context-free grammar, from the point of view of computational effectiveness it was much worse. As the above description implied, it was set up as a generative model, i.e. it told you how to produce a sentence starting from the symbol 'S'. Running the model in reverse to do sentence analysis turned out to be a computational nightmare, largely because transformations operate on trees, not strings of words, and so are highly non-deterministic when run backwards. For instance, the "equi-NP deletion" transformation deletes without trace the second occurrence of a co-referential noun phrase in certain structures, and it is impossible to run a deletion backwards if there is no clue as to what was deleted. Consequently, although some attempts have been made (e.g. Petrick [53]), parsers based on transformational grammar have not played a major role in natural language processing.

2.2.3. Augmented transition networks

Largely in response to the problems of transformational grammar, Bobrow and Fraser [4] proposed and Woods [75] subsequently developed a method of expressing a syntactic grammar that *was* computationally tractable and yet still could capture linguistic generalizations in a concise way, in many cases more concisely than transformational grammar itself. The formalism Woods developed was known as an augmented transition network or ATN. It consisted of a recursive transition network

[5] Although, significantly, a complete transformational grammar of English has never been produced.

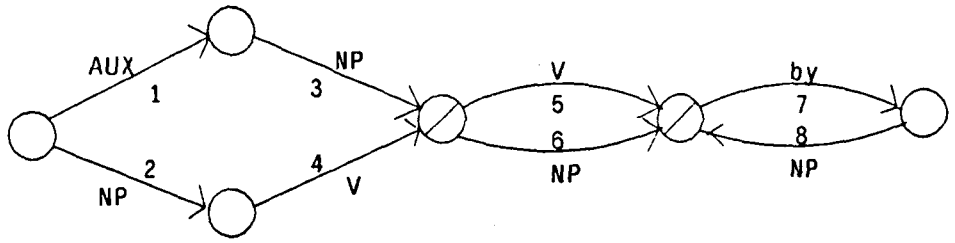

Figure 5: An example ATN

(formally equivalent in expressive power to a context-free grammar), augmented by a set of tests to be satisfied before an arc was traversed and a set of registers that could be used to save intermediate results or global state. An example ATN is shown in Figure 5. The network recognizes simple sentences with just a subject. verb and direct object in all combinations of active, passive, declarative, and interrogative. The symbols attached to the arcs show what constituent must be recognized to traverse the arc; AUX is an auxiliary verb (like "is" or "have"); NP is a noun phrase which is defined by another network in the same formalism as this one; V is a verb; and by is the word "by". The numbers on the arcs serve as indices into the following table which list the tests that must be true to traverse the arcs and the action that must be performed as the arc is traversed.

	TEST	PREDICATE
1	T	(SETR V *) (SETR TYPE 'QUESTION)
2	T	(SETR SUBJ *) (SETR TYPE 'DECLARATIVE)
3	(agrees * V)	(SETR SUBJ *)
4	(agrees SUBJ *)	(SETR V *)
5	(AND (GETF PPRT) (= V 'BE))	(SETR OBJ SUBJ) (SETR V *) (SETR AGFLAG T) (SETR SUBJ 'SOMEONE)
6	(TRANS V)	(SETR OBJ *)
7	AGFLAG	(SETR AGFLAG FALSE)
8	T	(SETR SUBJ *)

In this Lisp-like notation, "*" refers to the constituent just parsed, and SETR sets a register, whose name is specified by its first argument, to the value of its second argument. A concrete example of

the network in operation will make this clearer. Suppose we wanted to parse:

The rabbit nibbled the carrot.

We start at the leftmost node in the graph, and at the left of the sentence. Two arcs lead from that node, but only arc 2 is applicable since in the input we are not looking at the auxiliary verb required by arc 1, but are looking at a noun phrase, "the rabbit", as required by arc 2. We can see from the table that arc 2 has no additional test (indicated by T), so we traverse that link setting the SUBJ register to the thing just parsed, viz. "the rabbit", and the TYPE register to DECLARATIVE. We are now at a node with only one arc, arc 3, and that arc requires a verb. Fortunately, we are now looking at "nibbled" in the input, so we can try to traverse it. Arc 3 has an additional test requiring that *, i.e. the verb, agrees with what is in the subject register; this is the way agreements are enforced in an ATN. In our case, the agreement is correct and we can traverse the arc, setting the V register to the verb. The node we get to now has a line through it, indicating that this can be the end of the parse providing that there is no input left to consume, so "The rabbit nibbled." would be accepted here. In our example, there is another noun phrase, "the carrot", and so we follow arc 6, whose test requires the verb in the V register be transitive, which "nibbled" is. So we end up at another terminal node with no further input, and so the parse is completed successfully. The result of the parse is the setting of the four registers: SUBJ, TYPE, V, and OBJ, and these can be combined into a tree or whatever representation is desired.

A more interesting use of registers can be seen from the example:

The carrot was nibbled by the rabbit.

To parse the first three words, we traverse arcs 2 and 3 much as before, with the difference that now "the carrot" is in SUBJ and "was" is in V. Now we cannot take arc 6 because we are only up to "nibbled" in the input, but we can take arc 5 because "nibbled" is a verb. The test on arc 5 also requires "nibbled" to be a past participle, which it is, and the contents of V to be "be", and since "was" is a form of the verb to be, the test is satisfied. The action on arc 5 is interesting; it puts the contents of the SUBJ register in the OBJ register, overwrites the verb register with the past participle verb, sets a flag to true, and puts a placeholder "someone" in the SUBJ register. This corresponds to recognizing that the sentence is in passive form, and in our case makes "the carrot" the object and "nibbled" the verb. Now we reach "by" in the input and so can follow arc 7, which just requires the passive flag to be set; its only action is to turn this flag off, so that the arc cannot be traversed again. Finally, we get back to the terminal node via arc 8 which puts "the rabbit" in the SUBJ register. Note that the result of this parse is the same as the result of the first example. Now try yourself to follow the parses of:

Did the rabbit nibble the carrot?
Was the carrot nibbled by the rabbit?

These brief examples should give you some idea of the power of an ATN and of how its tests and registers can be used to capture the regularities of language in a concise and elegant way. Very large ATN grammars of several hundred nodes [76] have been developed that capture large subjects of English. However, ATNs also have several disadvantages:

- **complexity and non-modularity**: As the coverage of an ATN increases, so does its structural complexity. It becomes extremely difficult to modify or augment an existing ATN without causing large numbers of unforeseen side-effects. For instance, if another outgoing arc is added to a node with a large number[6] incoming arcs in order to handle an additional type of phrase that is a valid continuation of the parse represented by one of the incoming arcs, it could to lead spurious and incorrect parses when the node is reached via a different incoming arc.

- **fragility**: The current position in the network is a very important piece of state information for the operation of an ATN. If an input should be slightly ungrammatical, even by a single word, it is very hard to find the appropriate state to jump to that would enable the parse to continue, though see the work by Kwasny and Sondheimer [47] and Weischedel and Black [71] on dealing with such extra-grammaticality and the work on island-driven ATN parsing for speech input by Bates [77].

- **inefficiency through backtracking search**: Although the above examples are not complex enough to show it, the task of traversing an ATN is in general non-deterministic and requires search. The natural way to search an ATN is through backtracking. Because intermediate failures are not remembered in such a search, major inefficiencies can result through repetition of the same subparses arrived at through different paths through the network. Chart parsing techniques [43, 45, 29] were designed as alternatives to ATNs precisely to avoid these inefficiencies.

- **inefficiency through meaningless parses**: Normally the grammar of an ATN is purely syntactic and a complete syntactic parse is produced before any semantic interpretation is performed. In that situation, many spurious meaningless parses can be produced, especially if the grammar is large and comprehensive. To combat this, recent parsers [5] in the ATN tradition have tried to interpret each constituent as it was produced, thus preventing complete parses based on constituents that could be predicted to be nonsensical.

See [37, 39] for more discussion on the relative advantages and disadvantages of ATNs.

2.3. Semantic Grammars

Language analysis based on semantic grammars is just like syntactically-driven parsing except that in semantic grammars the categories used are defined semantically as well as or instead of syntactically. Thus instead of the category 'noun phrase' in a syntactic grammar, a semantic grammar might have the category, 'description of a ship', which is syntactically always a noun phrase, but has additional strong semantic constraints. Semantic grammars were invented by Burton [7] for

---

[6] ten or more is not uncommon in large realistic grammars

use in SOPHIE [6], a computer-aided instruction system for electronic circuit debugging, to deal with the problems of inefficiency due to the generation of syntactically correct, but meaningless, parses mentioned above for ATN-based syntactic grammars. The goal was to eliminate the production of meaningless parses by setting up the grammar so that only meaningful parses could be produced. To do this, it was necessary to categorize all the objects and actions that the SOPHIE system needed to parse to conduct a conversation in its domain of electronic circuitry, and then to construct the grammar so that, for instance, only a description of a switch could be the object of a "close" action. This technique, while retaining the fragility of an ATN, worked well to reduce parsing inefficiency. Because the relevant semantic categories were available at parse time, it also allowed semantic interpretation to proceed as the parse unfolded. However, the technique only works properly in restricted domains, like the one mentioned above, in which all objects and their relations can be categorized in advance, allowing a grammar to be built around the possible semantic relations. Semantic grammars are thus a technique useful only for applied natural language processing, not for general NLP.

For an example of how semantic grammars can be used, consider the following grammar definition in the formalism used by LIFER, a system for building semantic grammars developed by Hendrix [41].

```
S -> <present> the <attribute> of <ship>
<present> -> what is | [can you] tell me
<attribute> -> length | beam | class
<ship> -> the <shipname> | <classname> class ship
<shipname> -> kennedy | enterprise
<classname> -> kitty hawk | lafayette
```

An expanded version of this grammar was used for access to a database of information about US Navy ships in the LADDER [57] system, and even the above version is capable of recognizing such inputs as:

What is the length of the Kennedy?
Can you tell me the class of the Enterprise?
What is the length of Kitty Hawk class ships?

Since the definitions used by LIFER are similar to those used for context-free grammars, the reader should have no difficulty in seeing how these inputs could be recognized by the above grammar. In addition to defining a grammar, LIFER also allowed an interface builder to specify the interpretations to be produced from rules that were used in the recognition of an input. In the above case, this resulted in data base query language statements corresponding to the inputs being produced as a direct result of the recognition. The data base query language statements in effect took the place of a parse tree and so no separate semantic interpretation stage was required.

Note in the example above that not all the categories are specializations of pure semantic

categories; <present>, for instance, will parse several phrases, none of which fits into any standard grammatical category, and which differ from each other in their syntactic structure, including the number of verbs they contain. The ability to construct cross-grammatical categories like this allows a semantic grammar to incorporate some features of pattern-matching parsing. Also note how strongly directed the recognition is. The word, "class", for instance occurs in two quite different ways in the grammar: once as a ship attribute and the other as part of the second type of ship description. Thus in the (rather silly) question:

What is the class of Lafayette class ships?

the appropriate category for "class" would be used each time it appeared without considering its other role in the grammar. This directedness of recognition is also useful in building spelling correction into the recognition process. In an input like:

What is the legnth of the Kennedy?

the spelling of "legnth" need only be checked against the list of ship attributes, rather than the entire system vocabulary because a ship attribute is the only category that can appear at the place where the misspelling occurs. A final advantage of the strong top-down direction available through semantic grammars can be seen in LIFER's ellipsis mechanism which was intended to deal with input sequences such as:

What is the length of the Kennedy?
The beam?

Here the fact that "beam" and "length" are in the same semantic grammar category allows the second input to be interpreted as "What is the beam of the Kennedy?" rather than say "What is the length of the beam?". See Section 3.1 for more discussion on ellipsis mechanisms in general and that of LIFER in particular.

In addition to their numerous advantages for limited domain applications, semantic grammars have several disadvantages, chief of which is the requirement that a new grammar be developed for each new domain, since the semantic categories for each domain will be quite different. However, if the applications are similar (e.g. both include data base access), there will be many parts of the grammar (e.g. the basic framework for questions), that are the same. A related disadvantage is that semantic grammars tend to get large very quickly partly because of the repetition of similar constructions in different semantic categories. This makes non-toy semantic grammars quite hard to construct, and can result in very "spotty" kind of coverage of syntactic variation. For instance, if the possessive can be apostrophized in the description of a ship attribute (i.e. you can say "the Kennedy's length" as well as "the length of the Kennedy"), there is no inherent reason why it should also be apostrophized in the description of a attribute of a sailor (i.e. you might not be able to say "officer's rank" even though you can say "rank of an officer") because the two categories are in different parts of the grammar and

their recognition is unrelated.

Three approaches have been tried to resolve these problems. One is to go back to recognition by a syntactic grammar before semantic interpretation, but to try to intermix the semantic and syntactic components much more closely, so that every syntactic constituent is interpreted as soon as it is constructed. The RUS system [5] is an example of this approach. It provides a great improvement over a pure syntax first approach, but is still not as efficient as pure semantic grammars; it is also difficult to incorporate the pattern-matching possibilities of semantic grammars mentioned above.

An alternative approach, as exemplified by the TEAM system [34], is to focus in on a specific kind of application, in the case of TEAM, access to a relational database, and abstract out the linguistically common aspects of a semantic grammar for such an application. To build a specific interface, it is then only necessary to fill in a template, as it were, with the vocabulary and morphological variation required for a specific data base. This approach has the potential to produce highly efficient natural language interfaces, but at the cost of some expressive power.

The third approach is to combine the strengths of several parsing strategies, such as semantic grammars, syntactic transformations and pattern matching into a single system that maps structures into more canonical forms before attempting to use the full semantic grammar, thus allowing many redundant and unnecessary constructions to be eliminated [10, 11]. This multi-strategy approach has been implemented in the DYPAR system [14] and applied to data-base query, expert system command, and operating-system command interfaces.

2.4. Case-Frame Instantiation

A major development in computational linguistics was the inclusion of case-frame instantiation in the repertoire of effective parsing techniques. Case frames were introduced by the linguist Charles Fillmore in his seminal paper "A Case for Case" [28], and their computational import was quickly grasped by several researchers in natural language processing, including Simmons [67], Schank [59], and Riesbeck [54]. Case frame instantiation is one of the major parsing techniques under active research today. Its recursive nature, and its ability to combine bottom-up recognition of key constituents with top down instantiation of less structured constituents gives this method very useful computational properties.

2.4.1. What are case frames?

A case frame consists of a head concept and a set of roles, or subsidiary concepts, associated in a well-defined manner with the head concept. Initially, only sentential-level case frames were investigated, where the head consists of the main verb, and the cases include the "agent" that carries out the action, the "object" acted upon, the "location" in which the action takes place, etc. For instance, consider the sentence:

"In Elm Street, John broke a window with a hammer for Billy"

In simplified generic notation, the case frame corresponding to this sentence is:

```
[BREAK
  [case-frame
     agent: JOHN
     object: WINDOW
     instrument: HAMMER
     recipient:
     directive:
     locative: ELM STREET
     benefactive: BILLY
     co-agent: ]
  [modals
    time: past
    voice: active]]
```

In the notation above, cases, such as *agent* are written in small italics, and their fillers are in uppercase letters.

Case frames, as adopted in computational linguistics, differ markedly from simple, purely syntactic, parse trees. The relations between the head of the case frame and the individual cases are defined semantically, not syntactically. Hence, a noun in the subject position can fill the *agent* case, as in our example above, or it can fill an *object* case, as in "the window broke" (the window was not the agent that caused the breakage), or it can fill the *instrument* case, as in "the hammer broke the window". These are different semantic roles played by the same syntactic constituent, "subject". Since the purpose of a natural language interface is to extract the semantics of the input, it behooves the case frame representation to encode explicitly semantic differences in otherwise similar syntactic parse trees. Thus, parsing into case frames requires semantic knowledge, as well as syntactic information, as we shall see below.

Let us examine some other properties of case frames. In the example above, only some of the cases were instantiated. What of the other cases, such as *recipient* and *co-agent*? We shall get to examples that illustrate these momentarily. First, consider the meaning of each case, as outlined below:

```
[<HEAD VERB>
  [case-frame
     agent: <the active causal agent instigating the action>
     object: <the object upon which the action is done>
     instrument: <an instrument used to assist in the action>
     recipient: <the receiver of an action - often the indirect object>
     directive: <the target of a (usually physical) action>
     locative: <the location where the action takes place>
     benefactive: <the entity on whose behalf the action is taken>
     co-agent: <a secondary or assistant active agent>
   ]]
```

If instead of saying "John broke the window with a hammer", one were to say "John broke the window with Mary", Mary would fill the *co-agent* case. Presumably John did not swing Mary over his head and use her as a battering ram to shatter the window, much as he would use an instrument like a hammer or a tree branch. Since Mary is taking part in causing the action to happen, regardless of whether her action is independent of, or in support of John's action, she fills the *co-agent* case.

In order to illustrate the *directive* case, consider "John kicked the ball towards the goal" and "John flew the airplane to New York". In the former example "the goal" fills the *directive* case, and in the latter "New York" fills the same case, since both express the direction in which each respective action was performed. In some early formulations of case frames no distinction was made between *locative* and *directive*, but the need to encode explicitly stative versus dynamic information -- plus the need to represent sentences such as "In Yankee Stadium, John threw the ball at the catcher." that instantiate both cases -- led to the acceptance of two, semantically distinct, cases, one encoding global location, the other a local change in location.

The recipient case is filled by "Mary" in both of the following: "John gave Mary a ball" and "John gave a ball to Mary". Note that in this instance we have syntactically distinct sentences that map onto a unique semantic case frame representation, to wit:

```
[GIVE
  [caseframe
     agent: JOHN
     recipient: MARY
     object: BALL]
  ...]
```

2.4.2. Required, optional, and forbidden cases

Each case frame defines some required cases, some optional cases, and some forbidden cases. A required case is one that must be present in order for the verb to make sense. For instance, "break" requires the *object* case. A sentence is not complete without it (try constructing one), but no other case is required. "The window broke" is a complete, if not very informative, sentence. An optional

case is one which, if present, provides more information to the case-frame representation, but if absent, does not harm its semantic integrity. For instance *agent*, *co-agent*, and *locative* are optional cases of "break". Forbidden cases are those that cannot be present with the head verb. The *directive* and *recipient* cases are forbidden for the "break" case frame. (Again, try constructing a sentence with these cases using "break" as the head verb.)

2.4.3. Conceptual Dependency

It is often useful in natural language processing to employ a semantic representation which represents information in as canonical a manner as possible. In the ideal canonical representation, different ways of stating the same information would be represented identically, and propositions that encode similar information, would map into semantic encodings that highlighted the similarities while retaining the differences in an explicit manner. The best known attempt at a canonical semantic representation is the Conceptual Dependency (CD) formalism developed by Schank [59, 60, 62] as a *reductionistic* case frame representation for common action verbs. Essentially, it attempts to represent every action as a composition of one or more primitive actions, plus intermediate states and causal relations.

To use Shank's example, suppose we want to represent in a case frame notation: "John gave Mary a ball" and "Mary took a ball from John". These sentences differ syntactically, they differ in terms of verb selection, and they differ in how their cases are instantiated (e.g., "John" is the agent of the first sentence, and "Mary" of the second sentence). However, both sentences express the proposition that a ball was transferred from John to Mary, and, in both cases, we can infer that John had the ball before the action took place, that Mary has it after the action, and that John no longer has it after the action. The only significant difference is that in the first sentence, John performed the action, and in the latter Mary did so.

In Conceptual Dependency there is a primitive action called "ATRANS" (for Abstract TRANSfer of possession, control or ownership) that encodes the basic semantics of both of these verbs, and many more. The CD representation[7] of these sentences is:

---

[7] Some readers may be acquainted with Schank's complex notation of double and triple arrows. The present direct simplified notation is virtually isomorphic, somewhat clearer, and closer to the data structures used by most of the computer programs that parse into CD and other case frame representations.

```
[ATRANS                                  [ATRANS
  rel:  POSSESSION                         rel:  POSSESSION
  actor:  JOHN                             actor:  MARY
  object:  BALL                            object:  BALL
  source:  JOHN                            source:  JOHN
  recipient:  MARY]                        recipient:  MARY]

"John gave Mary a ball"                  "Mary took a ball from John"
```

These two structures are very simple to match against each other to determine precisely in what aspects the two propositions differ, and in what aspects they are identical. Moreover, inference rules associated with ATRANS can be invoked automatically when "give" and "take" are parsed into these structures. There are many more verbs that contain the ATRANS primitive (such as bequeath, donate, steal, sell, buy, appropriate, expropriate, etc.). Sometimes ATRANS is used in conjunction with other CD primitives that capture other aspects of the meaning. The verb "sell", for instance, involves two ATRANS primitives in mutual causation:

```
[ATRANS                                                  [ATRANS
  rel:  OWNERSHIP          CAUSE                         rel:  OWNERSHIP
  actor:  JOHN            -------->                      actor:  MARY
  object:  APPLE         <--------                       object:  25 CENTS
  source:  JOHN            CAUSE                         source:  MARY
  recipient:  MARY]                                      recipient: JOHN]

        "John sold an apple to Mary for 25 cents"
```

The cases used in conceptual dependency are similar but not identical to the set used originally in case grammars, although the basic ideas are the same. One refinement in CD was to separate *agent* into *actor* and *source*, as the two can be instantiated by different entities in the underlying semantic primitives. Other CD primitive actions include:

PTRANS	Physical transfer of location
MTRANS	Mental transfer of information
MBUILD	Create a new idea or conclusion from other information
INGEST	Bring any substance into the body
PROPEL	Apply a force to an object
ATTEND	Focus a sense organ (e.g., eyes, ears)

SPEAK Produce sounds of any sort

Later work [61] has extended this list to include social and other interpersonal actions.

2.4.4. Parsing into case frames

Our discussion of case frames thus far has focused on their structural properties, including parsimony and clarity of representation. Now we turn to the uses of case frames in parsing natural language, and in particular to certain parsing techniques available to parsers whose target representation is based on case frames. In essence, parsers built around case grammars help to combine bottom up recognition of structuring constituents with more focused top-down instantiation of less structured, more complex constituents. This essential property is evidenced in the example case frame recognition algorithm presented below.

Thus far we mentioned that case frames consist of a header and a collection of semantically defined cases. There is a bit more to it than that. Each case consists of a filler and a positional or a lexical marker. We saw examples of case fillers in the previous sections. A positional case marker says that the filler of the case occurs in a predefined location in the surface string. A lexical case marker says that the case filler is preceded by one of a small set of marker words (usually prepositions) in the surface string. For instance, consider the following input to a natural language interface to an operating system:

"Copy the fortran files from the system library to my directory"

"Copy" is the case header, the *object* case is marked positionally as the noun phrase occupying the simple direct object position (i.e., the first noun phrase to the right of the verb that is not preceded by a preposition). The filler of the object case is constrained semantically to be some information structure in a computer. Hence, the parser knows where in the input to search for the filler of the object case, and moreover knows what to expect in that position (a noun phrase denoting an information structure, like a file or directory in a computer). The *source* case is marked lexically by the preposition "from" and the *recipient* case is marked by the preposition "to". Both case fillers are constrained to be noun phrases denoting information repositories in the computer (directories, tapes, etc.). More explicitly, the case frame information available to the parser is:

```
[COPY <header-pattern>
   [object:
       marker: (POSITIONAL DIRECT-OBJECT)
       filler: <information-structure>]
   [source:
       marker: (LEXICAL <from-marker>)
       filler: <information-repository> | <output-device>]
   [destination:
       marker: (LEXICAL <to-maker>)
       filler: <information-repository> | <input-device>]
   ]

Where:
 <header-pattern> -> copy | transfer | move | ...
 <from-maker> -> from | in
 <to-marker> -> to | into | onto

plus patterns or NP-level case frames to recognize output-devices,
input-devices, information structures, and information-repositories
```

A typical case-frame parsing algorithm that operates on this case frame data structure could be summarized as follows:

1. For each case frame in the grammar, attempt an unanchored match of the header pattern against the input string. If none succeed, the input is unparsable by the grammar.[8] If one or more matches are found, perform the following steps for each case header, and the one(s) that account for the entire input are the possible parses of the input string.

2. Retrieve the case frame indexed by the recognized case header.

3. Attempt to recognize each required case, as follows:

   - If the case is marked lexically, do an unanchored match for the case marker (a very simple one-or-two word pattern), and if that succeeds, perform the more complex recognition of the case filler by anchored match to the right of the case marker, or by a more complex parsing strategy (such as recognizing an embedded case frame starting at that location in the input).

   - If the case is marked positionally, do an anchored match of the case filler (or again a more complex recognition strategy) starting at the designated point in the input string.

   - If the case maker can be marked either way, search first for the lexical marker, and failing that attempt to recognize it positionally. For instance, the *recipient* case in GIVE can be marked by the word "to" (or "unto", etc.) or it can appear positionally in the indirect object location ("John gave an apple to Mary" vs "John gave Mary an apple").

---

[8] An *unanchored match* is the process of searching for a particular pattern anywhere in the input, as opposed to an *anchored match*, where the match is attempted only starting at a predefined position in the input string.

If one or more required cases are not recognized, return an error condition. This signifies a possible ellipsis, incorrect selection of the case frame, ill-formed user input, or insufficient grammatical coverage. The following sections address issues of robust recovery from ill-formed user input.

4. Attempt to recognize all the optional cases by applying the same method used to parse the required ones. If some are not recognized, however, do not generate error conditions.

5. If after all the required and optional cases have been processed, and there is remaining input, generate a potential error condition denoting spurious input, insufficient coverage, or garbled or ill-formed input that may be recognized by more flexible parsing strategies.

As the case frame is parsed, the input segments recognized as case fillers are processed and stored as the value of the corresponding cases in the case frame. A partially-instantiated case frame can serve to guide error-correction processes or to formulate focused queries to the user [10, 35, 38]. The initial case frame selection phase can be speeded up by indexing the case-header patterns by the words they contain and recognizing them in a pure bottom-up manner. This bottom-up index-based process is computationally effective if there are very many case frames, and each case header consists of a relatively simple pattern. Otherwise, the top-down un-anchored pattern match is sufficiently efficient (few case frames), or both processes require substantial computation (large numbers of case frames with complex header patterns).

Case-frame instantiation can be applied recursively to parse relative clauses or any other linguistic structures that can be expressed as case frames. Noun phrases with post-nominal modifiers (i.e., trailing propositional phrases that modify the main noun phrase), for instance, can be encoded and recognized by an extension of the sentential-level case-frame instantiation algorithm presented above. Moreover, case-frame instantiation works in concert with semantic grammars or patterns used to recognize any subconstituents such as case markers represented as non-terminal nodes in a grammar.

The advantages of case frame instantiation over other parsing techniques can be summarized as follows:

- Case frames combine bottom-up recognition of simple structuring constituents, such as case headers and case markers, with top-down recognition of semantically more complex, but syntactically less significant case fillers. The differential treatment of different constituents provides more efficient parsing in general, allows for ellipsis resolution, and makes possible some forms of error recovery, as discussed below.

- Case frames combine syntax and semantics. Positional and case-marker information is used in concert with semantic recognition of case fillers, thus reducing (though certainly not eliminating) structural and lexical ambiguity.

- Case frames are a fairly convenient representation for back-end systems to use. In contrast, parse trees must first be interpreted semantically, and subsequently transformed into a representation more convenient for other modules in the system.

2.5. Robust Parsing

Any natural language interface which is used in a practical application with a multitude of users must be able to handle input that is outside its grammar or expectations in various ways. When people use language spontaneously, whether in spoken or written form, they inevitably make mistakes and these will account for many of the extragrammatical utterances that a natural language interface will receive. Given the present limited state of NLP, a natural language interface must also be prepared for input that is, as far as the user is concerned, perfectly correct, but which the parser cannot recognize because of its own limited competence. Some types of extragrammatical utterances (see [37, 39] for more complete accounts) are listed below with example utterances that might be encountered by an interface to a data base of college courses in which the courses are identified by a the name of a department followed by a number.

- spelling errors

  tarnsfer Jim Smith from Econoics 237 too Mathematics 156

  Note that some spelling errors can result in different correctly spelt words (e.g. "too").

- novel words

  transfer Smith out of Economics 237 to Basketwork 100

  Here we suppose that "out of" is not listed as a (multiword) preposition corresponding to the source case marker of transfer, and that "Basketwork" is not in the interface's dictionary of department names.

- spurious phrases

  please enroll Smith if that's possible in I think Economics 237

- ellipsis or other fragmentary utterances

  also Physics 314

  This might be the a follow-up input to the previous one.

- unusual word order

  in Economics 237 Jim Smith enroll

- missing words

  enroll Smith Economics 237

  Here the "in" is missing, but the meaning is still perfectly clear.

Unless a natural language interface can deal with problems in these classes easily, it will appear very uncooperative and stupid to its users, who will tend either not to use it if they have that choice, or to

use it with a high-level of frustration. We will examine techniques available to deal with some of the above deviations from grammaticality in more detail.

Spelling errors are the most common and normally the most easily corrected of all grammatical deviations. The usual basic approach when a word is found to be outside the vocabulary of a natural language interface is to compare the word against a set of known words and substitute the word (or words) from that list found to be closest to the unknown word according to some metric and subject to some threshold of closeness. We do not have time here to go into the methods of comparison, but clearly, the process will be made more efficient and less prone to error by shortening the list of words against which to compare the unknown word. For this reason, methods of language analysis, such as semantic grammars and case-frame instantiation, that are able to apply strong top-down constraints to their recognition are at a significant advantage when it comes to spelling correction. For instance, in:

tarnsfer Jim Smith from Econoics 237 too Mathematics 156

a system based on case-frame instantiation such as we examined in Section 2.4 need only compare "Econoics" against its list of department names rather than against its whole vocabulary. This ability is particularly important in the case of "too" in this example, because "too" is a real word that might well be in the system's vocabulary, and without the strong prediction that it should be a preposition marking a case of "transfer", the system would be unable to correct it (a match against the whole vocabulary would make "too" the best match) or even notice that it is misspelt.

Whereas spelling correction can be dealt with at the lexical level, other forms of grammatical deviation require modification to a NLP system's grammatical expectations. The way in which this can be accomplished differs markedly by approach. In pattern matching, for instance, the obvious approach is partial pattern matching as attempted in the FlexP system [37]. Patterns are deemed to match partially if most but not all their elements actually do match the input. Clearly, this can be useful for missing or extra words, but is not useful in the case of unusual word order. Moreover, in practice it turns out that some elements of a pattern are more important than others, and unless allowance is made for these differences, it is difficult to decide exactly how much of a pattern needs to match before the pattern as a whole can be declared to have matched.

Dealing with grammatical deviation in an ATN-based system turns out to be extremely difficult. The current position in the network is a very important piece of state information for the operation of an ATN. If an input should be slightly ungrammatical, even by a single word, it is very hard to find the appropriate state to jump to that would enable the parse to continue. This assumes, moreover, that it is possible to determine exactly where the input has departed from the grammar's expectations. The

backtracking search used with most ATNs can make this difficult. Work by Weischedel and Black [71] has dealt with extragrammaticality caused by incorrect agreements that can be resolved by relaxing the predicates on ATN arcs, and Kwasny and Sondheimer [47] have looked into adding extra arcs to ATNs on a dynamic basis to make the grammar fit the input. Earlier work on speech parsing [77] also tried to use ATNs in an island-driven mode.

A more recent development in robust parsing by Carbonell and Hayes [12, 38] uses a *construction specific* approach that fits in well with semantic grammars and case frame instantiation. The basic idea is to tailor parsing strategies to specific construction types; this not only results in efficient parsing of grammatical input, but also permits built-in recovery strategies that exploit the characteristics of the particular construction type. For instance, the following simple recovery strategy works quite well for simple imperative case frames:

```
skip over unexpected input until a case marker is found;
parse skipped segments against unfilled cases.
```

If this strategy is applied to:

transfer Economics 247 to Physics 317 Smith

"Economics 247" and "Smith" will initially be skipped over, with "to Physics 317" being correctly parsed since "to" is a valid case marker. Then the skipped segments will be correctly parsed against the unfilled cases, "source-course" and "student", respectively, leading to a parse identical to that for:

transfer Smith from Economics 247 to Physics 317

Such methods of robust parsing are under active investigation at the moment with the chief outstanding problem being the coordination of multiple, independent, construction-specific parsing strategies on the same input.

3. Dialogue Phenomena

In addition to recognizing individual sentences, the problem of interactive communication through natural language, be it communication between man and machine or communication between two people, entails discourse phenomena that transcend individual sentences.

- **Anaphora** — Pronouns and other anaphoric references (words like "it", "that" or "one") refer to concepts described previously in a dialogue. Anaphoric resolution entails identifying the referents of these place-holder words. Interactive dialogues invite the use of anaphora, much more than simpler data base query situations. Therefore, as natural language interfaces increase in complexity and expand their domain of application, anaphoric resolution becomes an increasingly important problem.

- **Definite noun phrases** — Noun phrases often serve as another type of anaphoric reference by referring to previously mentioned concepts, much like the less specific

anaphors do. Usually such phrases are flagged by a definite pronoun (e.g., "the"). As Grosz [32] noted, resolving the referent of definite noun phrases or any other anaphors often requires an understanding of the planning structure underlying cooperative discourse.

- **Ellipsis** — People often use sentence fragments to express a complete proposition. These terse utterances must be filled out in the context of the dialogue. Sentential level ellipsis has long been recognized as ubiquitous in discourse. However, *semantic ellipsis*, where ellipsis occurs through semantically incomplete propositions rather than through syntactically incomplete structures, is also an important phenomenon. The ellipsis resolution method presented in Section 3.1 addresses both kinds of ellipsis.

- **Extragrammatical utterances** — Interjections, dropped articles, false starts, misspellings, and other forms of grammatical deviance abound. Developing robust parsing techniques that tolerate errors has been the focus of much recent work [12, 38, 40, 46, 71], as discussed in the preceding section.

- **Meta-linguistic utterances** — Intra-sentential metalanguage has been investigated to some degree [42, 56], but its more common inter-sentential counterpart has received little attention [17]. However, utterances about other utterances (e.g., corrections of previous commands, such as "I meant to type X instead" or "I should have said ...") are not infrequent, and an initial stab is being made at this problem [36]. Note that it is a cognitively less demanding task for a user to correct a previous utterance than to repeat an explicit sequence of commands (or worse yet, to detect and undo explicitly each and every unwanted consequence of a mistaken command).

- **Indirect speech acts** — Occasionally users of natural language interfaces will resort to indirect speech acts [2, 52, 65], especially in connection with inter-sentential metalanguage or by stating a desired state of affairs and expecting the system to supply the sequence of actions necessary to achieve that state.

Our own empirical studies suggest that users of natural language interfaces avail themselves of discourse phenomena whenever such devices help in formulating short, succinct linguistic expressions over lengthier, more explicit ones. This observation is summarized as follows:

> **Terseness principle:** *Users of natural language interfaces insist on being as terse as possible, independent of task, communication media, typing ability, or instructions to the contrary, without sacrificing the flexibility of expression inherent in natural language communication.*[9]

[9] This principle may be viewed as a surprisingly strong form of Grice's maxim of brevity [31].

3.1. Case-Frame Ellipsis Resolution

In order to illustrate the ubiquity of ellipsis in interactive dialogues through a natural language interface, let us look at the XCALIBUR project, whose objective is to provide flexible natural language access (comprehension and generation) to the XSEL expert system [50]. XSEL, the Digital Equipment Corporation's automated salesman's assistant, advises on selection of appropriate VAX components and produces a sales order for automatic configuration by the R1 system [49]. Part of the XSEL task is to provide the user with information about DEC components, hence subsuming the data-base query task. However, unlike a pure data base query system, an expert system interface must also interpret commands, understand assertions of new information, and carry out task-oriented dialogues (such as those discussed by Grosz [32]). XCALIBUR, in particular, deals with commands to modify an order, as well as information requests pertaining to its present task or its data base of VAX component parts. In the near future it should process clarificational dialogues when the underlying expert system (i.e. XSEL) requires additional information or advice, as illustrated in the sample dialogue below:

>What is the largest 11780 fixed disk under $40,000?
The rp07-aa is a 516 MB fixed pack disk that costs $38,000.
>The largest under $50,000?
The rp07-aa.
>Add two rp07-aa disks to my order.
Line item 1 added: (2 rp07-aa)
>Add a printer with graphics capability
fixed or changeable font?
>fixed font
lines per minute?
>make it at least 200, upper/lowercase.
Ok. Line item 2 added: (1 lxy11-sy)
>Tell me about the lxy11
The lxy11 is a 240 l/m line printer with plotting capabilities.

For details of the XCALIBUR interface, the reader is referred to [14, 15, 18]. Here we focus only on illustrating the case-frame ellipsis resolution method.

The XCALIBUR system handles ellipsis at the case-frame level. Its coverage appears to be a superset of the LIFER/LADDER system [41, 57] and the PLANES ellipsis module [70]. Although it handles most of the ellipsed utterances we encountered, it is not meant to be a general linguistic solution to the ellipsis phenomenon. The following examples are illustrative of the kind of sentence fragments the current case-frame method handles. For brevity, assume that each sentence fragment occurs immediately following the initial query below.

INITIAL QUERY: "What is the price of the three largest single port fixed media disks?"

"Speed?"
"Two smallest?"
"How about the price of the two smallest?"
"also the smallest with dual ports"
"Speed with two ports?"
"Disk with two ports."

In the representative examples above, punctuation is of no help, and pure syntax is of very limited utility. For instance, the last three phrases are syntactically similar (indeed, the last two are indistinguishable), but each requires that a different substitution be made on parse of the preceding query.

Ellipsis is resolved differently in the presence or absence of strong discourse expectations. In the former case, the discourse expectation rules are tested first, and, if they fail to resolve the sentence fragment, the contextual substitution rules are tried. If there are no strong discourse expectations, the contextual substitution rules are invoked directly.

Exemplary discourse expectation rule:

```
IF:   The system generated a query for confirmation or
      disconfirmation of a proposed value of a filler
      of a case in a case frame in focus,
THEN: EXPECT one or more of the following:
        1) A confirmation or disconfirmation pattern.
        2) A different but semantically permissible filler
           of the case frame in question (optionally repeating
           the attribute or providing the case marker).
        3) A comparative or evaluative pattern.
        4) A query for possible fillers or constraints on
           possible fillers of the case in question.
           [If this expectation is confirmed, a sub-dialogue
            is entered, where previously focused entities
            remain in focus.]
```

The following dialogue fragment, presented without further commentary, illustrates how these expectations come into play in a focused dialogue:

>Add a line printer with graphics capabilities.

Is 150 lines per minute acceptable?

>No, 320 is better	*Expectations 1, 2 & 3*
(or) other options for the speed?	*Expectation 4*
(or) Too slow, try 300 or faster	*Expectations 2 & 3*

The utterance "try 300 or faster" is syntactically a complete sentence, but semantically it is just as fragmentary as the previous utterances. The strong discourse expectations, however, suggest that it be processed in the same manner as syntactically incomplete utterances, since it satisfies the expectations of the interactive task. The terseness principle operates at all levels: syntactic, semantic

and pragmatic.

The contextual substitution rules exploit the case-frame representation of queries and commands discussed in the previous section. The scope of these rules, however, is limited to the last user interaction of appropriate type in the dialogue focus, as illustrated below. The rules search the ellipsed fragment for case fillers (or case markers and filler pairs) to substitute for corresponding cases in the parse of the previous input. Substitution can occur at a top level (sentential) case frame or in embedded (relative-clause or noun-phrase) case frames. This process resolves ellipses such as:

>What is the size of the 3 largest single port fixed media disks?
>And the price and speed?

and:

>What is the size of the 3 largest single port fixed media disks?
>disks with two ports?

Note that it is impossible to resolve this kind of ellipsis in a general manner if the previous query is stored verbatim or as a semantic-grammar parse tree. "Disks with two ports" would at best correspond to some **<disk-descriptor>** non-terminal, and hence, according to the LIFER algorithm [41, 57], would replace the entire phrase "single port fixed media disks" that corresponded to **<disk-descriptor>** in the parse of the original query. However, an informal poll of potential users suggests that the preferred interpretation of the ellipsis retains the previous information in the original query. The ellipsis resolution process, therefore, requires a finer grain substitution method than simply inserting the highest level non-terminals in the ellipsed input in place of the matching non-terminals in the parse tree of the previous utterance.

Taking advantage of the fact that a case frame analysis of a sentence or object description captures the meaningful semantic relations among its constituents in a canonical manner, a partially instantiated nominal case frame can be merged with the previous case frame as follows:

- Substitute any cases instantiated in the original query that the ellipsis specifically overrides. For instance "with two ports" overrides "single port" in our example, as both entail different values of the same case filler, regardless of their different syntactic roles. ("Single port" in the original query is an adjectival construction, whereas "with two ports" is a post-nominal modifier in the ellipsed fragment.)

- Retain any cases in the original parse that are not explicitly contradicted by new information in the ellipsed fragment. For instance, "fixed media" is retained as part of the disk description, as are all the sentential-level cases in the original query, such as the quantity specifier and the projection attribute of the query ("size").

- Add cases of a case frame in the query that are not instantiated therein, but are specified in the ellipsed fragment. For instance, the "fixed head" descriptor is added as the media case of the disk nominal case frame in resolving the ellipsed fragment in the following

example:

>Which disks are configurable on a VAX 11-780?
>Any configurable fixed head disks?

- In the event that a new case frame is mentioned in the ellipsed fragment, wholesale substitution occurs, much like in the semantic grammar approach. For instance, if after the last example one were to ask "How about tape drives?", the substitution would replace "fixed head disks" with "tape drives", rather than replacing only "disks" and producing the phrase "fixed head tape drives", which is meaningless in the current domain. In these instances the semantic relations captured in a case frame representation and not in a semantic grammar parse tree prove immaterial.

The key to case-frame ellipsis resolution is matching corresponding cases, rather than surface strings, syntactic structures, or non-canonical representations. While correctly instantiating a sentential or nominal case frame in the parsing process requires semantic knowledge, some of which can be rather domain specific, once the parse is attained, the resulting canonical representation, encoding appropriate semantic relations, can and should be exploited to provide the system with additional functionality such as the present ellipsis resolution method. For more details and examples of the rules that perform case-frame substitution, see the XCALIBUR report [14].

3.2. More complex phenomena

In addition to ellipsis and anaphora, there are more complex phenomena that must be addressed if one is to understand and simulate human discourse. This type of deeper understanding has not yet been incorporated into practical natural language interfaces, not least because much more basic abilities are still not dealt with completely adequately. However, as natural language interfaces increase in sophistication (as they surely will), these more complex phenomena will need to be dealt with, so as a final topic in this tutorial, we will look at some examples of these more esoteric discourse phenomena.

3.2.1. Goal-determination inference

The interpretation of an utterance may depend on the inferred conversational goals of the speaker. Consider the following set of examples, in which the same utterance spoken in somewhat different contexts elicits radically different responses. These responses depend on the interpretation of the initial utterance, in which the attribution of goals to the speaker plays a dominant role.

`Passer-by:` *Do you know how to get to Elm Street?*

`Person on the street corner:`
Walk towards that tall building and Elm Street is the fifth or sixth on your left.

The passer-by's question was quite naturally interpreted as an indirect speech act, since the information sought (and given) was not whether the knowledge of getting to Elm Street was present,

but rather how actually to get there. Lest the mistaken impression be given that it is a simple matter to identify indirect speech acts computationally, consider the following variant to our example:

Passer-by: *Do you know how to get to Elm Street?*

Person reading a street map and holding an envelope with an Elm Street address on it:
No, I haven't found it; could you help me?

In the second example, the listener infers that the goal of the passer-by is to render assistance, and therefore the initial utterance is interpreted as a direct query of the knowledge state of the listener, in order to know whether assistance is required. Hence, the passer-by's question is not an indirect speech act in this example.

Nor is the task of the interpreter of such utterances only to extract a binary decision on the presence of a speech act from goal expectations. The selection of *which* indirect speech act is meant often rests on contextual attribution of different goals to the speaker. Consider, for instance, the following contextual variant of our previous example:

Passer-by: *Do you know how to get to Elm Street?*

Waiting cabbie: *Sure, hop in. How far up Elm Street are you going?*

In this example, the cabbie interpreted the goal of the passer-by as wanting a ride to an Elm Street location. Making sure the cabbie knows the destination is merely instrumental to the inferred goal. The social relation between a cabbie and a (potential) customer is largely responsible for triggering the goal attribution. Thus, the passer-by's utterance in this example is also also interpreted as an indirect speech act, but a different one from the first example (i.e., wanting to be driven to the destination vs. wanting to know how to navigate to the destination). In summary, three totally different speech acts are attributed to identical utterances as a function of different goals inferred from contextual information.[10]

Example	***Speech act***
1) Original example	Indirect information request
2) Map reader	Direct information request
3) Cabbie example	Indirect action request

[10]For additional discussion of goal-determination inferences in discourse comprehension see [2, 16, 29].

3.2.2. Social role constraints

The relative social roles of the discourse participants affects their interpretation of utterances as illustrated below:

Army General: *I want a juicy Hamburger.*
Aide: *Yes sir!*

Child: *I want a juicy Hamburger.*
Mother: *Not today, perhaps tomorrow for lunch.*

Prisoner 1: *I want a juicy hamburger.*
Prisoner 2: *Yeah, me too. All the food here tastes like cardboard.*

Clearly, the interpretation of the phrase "I want a juicy hamburger" differs in each example with no context present beyond the differing social roles of the participants, and their consequent potential for action. In the first example a direct order is inferred, in the second a request, and in the third only a general assertion of a (presumably unattainable) goal. Therefore, comprehending a dialogue rests critically on knowledge of social roles [16, 33]. Moreover, social role constraints provide part of the setting essential in making goal attributions, and therefore impinge (albeit indirectly) upon goal determination inferences discussed in the previous section. In unconstrained discourse there is strong interaction between goal expectations, social role constraints, indirect speech acts, and meta-language utterance interpretation.

4. Conclusion

This tutorial has presented a brief overview of the current state of the art of applied natural language processing — the business of making practical computer systems that communicate with their users through natural language. We have described the way this style of natural language processing differs from the more general open-ended kind also studied in AI and from the approach to natural language in Linguistics and Cognitive Psychology. As we have seen, practical natural language interfaces can currently only be constructed to perform limited tasks within restricted domains, and we have examined and compared the various techniques that have been employed to construct such interfaces. Further details on any of the systems or techniques we have described can, of course, be obtained by following the large set of references we have provided. A reader with a more general desire for further information may be particularly interested in [20, 58, 74], and a reader with a desire to see some implementation details of systems illustrative of the cognitive simulation approach may wish to look at [62] which includes unusually complete descriptions of a small number of natural language processing systems.

References

1. Allen, J. F. *A Plan Based Approach to Speech Act Recognition.* Ph.D. Th., University of Toronto, 1979.

2. Allen, J. F. and Perrault, C. R. "Analyzing Intention in Utterances." *Artificial Intelligence 15*, 3 (1980), 143-178.

3. Anderson, J. R.. *Language, Memory, and Thought.* Lawrence Erlbaum, 1976.

4. Bobrow, D. G. and Fraser, J. B. An Augmented State Transition Network Analysis Procedure. Proc. Int. Jt. Conf. on Artificial Intelligence, Washington, D. C., 1969, pp. 557-567.

5. Bobrow, R. J. The RUS System. BBN Report 3878, Bolt, Beranek, and Newman, 1978.

6. Brown, J. S. and Burton, R. R. Multiple Representations of Knowledge for Tutorial Reasoning. In *Representation and Understanding*, Bobrow, D. G. and Collins, A., Ed.,Academic Press, New York, 1975, pp. 311-349.

7. Burton, R. R. Semantic Grammar: An Engineering Technique for Constructing Natural Language Understanding Systems. BBN Report 3453, Bolt, Beranek, and Newman, Inc., Cambridge, Mass., December, 1976.

8. Carbonell, J. R. Mixed-Initiative Man-Computer Dialogues. Tech. Rept. 1970, Bolt, Beranek, and Newman, Inc., Cambridge, Mass., 1971.

9. Carbonell, J. G. *Subjective Understanding: Computer Models of Belief Systems.* Ph.D. Th., Yale University, 1979.

10. Carbonell, J. G. and Hayes, P. H. Robust Parsing Using Multiple Construction-Specific Strategies. In *Natural Language Processing*, L. Bolc, Ed.,(publication forthcoming), 1983.

11. Carbonell, J. G. Towards a Robust, Task-Oriented Natural Language Interface. Workshop/Symposium on Human Computer Interaction, Georgia Tech Information Sciences, March, 1981.

12. Carbonell, J. G. and Hayes, P. J. Dynamic Strategy Selection in Flexible Parsing. Proc. of 19th Annual Meeting of the Assoc. for Comput. Ling., Stanford University, June, 1981, pp. 143-147.

13. Carbonell, J. G., Boggs, W. M., Mauldin, M. L. and Anick, P. G. The XCALIBUR Project, A Natural Language Interface to Expert Systems. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 1983.

14. Carbonell, J. G., Boggs, W. M., Mauldin, M. L. and Anick, P. G. XCALIBUR Progress Report # 1: First Steps Towards an Integrated Natural Language Interface. Carnegie-Mellon University Computer Science Department, 1983.

15. Carbonell, J. G., Larkin, J. H. and Reif, F. Towards a General Scientific Reasoning Engine. Carnegie-Mellon University Computer Science Department, 1983. CIP # 445

16. Carbonell, J. G.. *Subjective Understanding: Computer Models of Belief Systems.* Ann Arbor, MI: UMI research press, 1981.

17. Carbonell, J. G. Beyond Speech Acts: Meta-Language Utterances, Social Roles, and Goal Hierarchies. Preprints of the Workshop on Discourse Processes, Marseilles, France, 1982.

18. Carbonell, J. G. Discourse Pragmatics in Task-Oriented Natural Language Interfaces. Proceedings of the 21st annual meeting of the Association for Computational Linguistics, ACL-83, 1983.

19. Charniak, E. C. Toward a Model of Children's Story Comprehension. Tech. Rept. TR-266, MIT AI Lab, Cambridge, Mass., 1972.

20. Charniak, E. and Wilks, Y. (Eds.). *Computational Semantics.* North-Holland, 1976.

21. Chomsky, N.. *Syntactic Structures.* Mouton, The Hague, 1957.

22. Chomsky, N.. *Aspects of the Theory of Syntax.* MIT Press, 1965.

23. Cohen, P. R. and Perrault, C. R. "Elements of a Plan-Based Theory of Speech Acts." *Cognitive Science 3* (1979), 177-212.

24. Colby, K. M. Simulations of Belief Systems. In *Computer Models of Thought and Language*, Schank, R. C. and Colby, K. M., Ed.,Freeman, San Francisco, 1973, pp. 251-286.

25. Cullingford, R. *Script Application: Computer Understanding of Newspaper Stories.* Ph.D. Th., Computer Science Dept., Yale University, 1978.

26. Dejong, G. *Skimming Stories in Real-Time.* Ph.D. Th., Computer Science Dept., Yale University, 1979.

27. Earley, J. "An efficient context-free parsing algorithm." *Comm. ACM 13*, 2 (1970), 94-102.

28. Fillmore, C. The Case for Case. In *Universals in Linguistic Theory*, Bach and Harms, Ed.,Holt, Rinehart, and Winston, New York, 1968, pp. 1-90.

29. Frederking, R. A Rule-Based Conversation Participant. Proceedings of the 19th Meeting of the Association for Computational Linguisitcs, ACL-81, 1981.

30. Gazdar, G. Phrase Structure Grammars and Natural Language. Proc. Eighth Int. Jt. Conf. on Artificial Intelligence, Karlsruhe, August, 1983, pp. 556-565.

31. Grice, H. P. Conversational Postulates. In *Explorations in Cognition*, D. A. Norman and D. E. Rumelhart, Eds., Freeman, San Francisco, 1975.

32. Grosz, B. J. The Representation and Use of Focus in a System for Understanding Dialogues. Proc. Fifth Int. Jt. Conf. on Artificial Intelligence, MIT, 1977, pp. 67-76.

33. Grosz, B. J. Utterance and Objective: Issues in Natural Language Communication. Proceedings of the Sixth International Joint Conference on Artificial Intelligence, IJCAI-79, 1979, pp. 1067-1076.

34. Grosz, B. J. TEAM: A Transportable Natural Language Interface System. Proc. Conf. on Applied Natural Language Processing, Santa Monica, February, 1983.

35. Hayes P. J. A Construction Specific Approach to Focused Interaction in Flexible Parsing. Proc. of 19th Annual Meeting of the Assoc. for Comput. Ling., Stanford University, June, 1981, pp. 149-152.

36. Hayes, P. J. and Carbonell, J. G. A Framework for Processing Corrections in Task-Oriented Dialogs. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 1983.

37. Hayes, P. J. and Mouradian, G. V. "Flexible Parsing." *American Journal of Computational Linguistics 7*, 4 (1981), 232-241.

38. Hayes, P. J. and Carbonell, J. G. Multi-Strategy Construction-Specific Parsing for Flexible Data Base Query and Update. Proc. Seventh Int. Jt. Conf. on Artificial Intelligence, Univ. of British Columbia, Vancouver, August, 1981, pp. 432-439.

39. Hayes, P. J. and Reddy, D. R. "Steps Toward Graceful Interaction in Spoken and Written Man-Machine Communication." *International Journal of Man-Machine Studies 18* (1983).

40. Hayes, P. J. and Carbonell, J. G. Multi Strategy Parsing and its Role in Robust Man-Machine Communication. Tech. Rept. CMU-CS-81-118, Carnegie-Mellon University Computer Science Department, May, 1981.

41. Hendrix, G. G. Human Engineering for Applied Natural Language Processing. Proc. Fifth Int. Jt. Conf. on Artificial Intelligence, MIT, 1977, pp. 183-191.

42. Joshi, A. K. Use (or Abuse) of Metalinguistic Devices. Unpublished Manuscript

43. Kaplan, R. M. A General Syntactic Processor. In *Natural Language Processing*, Rustin, R., Ed.,Algorithmics Press, New York, 1973, pp. 193-241.

44. Kaplan, S. J. *Cooperative Responses from a Portable Natural Language Data Base Query System.* Ph.D. Th., Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia, 1979.

45. Kay, M. The MIND System. In *Natural Language Processing*, Rustin, R., Ed.,Algorithmics Press, New York, 1973, pp. 155-188.

46. Kwasny, S. C. and Sondheimer, N. K. Ungrammaticality and Extragrammaticality in Natural Language Understanding Systems. Proceedings of the 17th Meeting of the Association for Computational Linguistics, ACL-79, 1979, pp. 19-23.

47. Kwasny, S. C. and Sondheimer, N. K. "Relaxation Techniques for Parsing Grammatically Ill-Formed Input in Natural Language Understanding Systems." *American Journal of Computational Linguistics 7*, 2 (1981), 99-108.

48. Marcus, M. A.. *A Theory of Syntactic Recognition for Natural Language.* MIT Press, Cambridge, Mass., 1980.

49. McDermott, J. R1: A Rule-Based Configurer of Computer Systems. Carnegie-Mellon University Computer Science Department, 1980.

50. McDermott, J. XSEL: A Computer Salesperson's Assistant. In *Machine Intelligence 10*, Hayes, J., Michie, D. and Pao, Y-H., Eds., Chichester UK: Ellis Horwood Ltd., 1982", pp. 325-337.

51. Parkison, R. C., Colby, K. M., and Faught, W. S. "Conversational Language Comprehension Using Integrated Pattern-Matching and Parsing." *Artificial Intelligence 9* (1977), 111-134.

52. Perrault, C. R., Allen, J. F. and Cohen, P. R. Speech Acts as a Basis for Understanding Dialog Coherence. Proceedings of the Second Conference on Theoretical Issues in Natural Language Processing, 1978.

53. Petrick, S.R. *A Recognition Procedure for Transformational Grammars.* Ph.D. Th., Dept. of Modern Languages, MIT, Cambridge, Mass., 1965.

54. Riesbeck, C. Conceptual Analysis. In *Conceptual Information Processing*, R. C. Schank, Ed.,Amsterdam: North-Holland, 1975, ch. 4, pp. 83-156.

55. Riesbeck, C. K. and Schank, R. C. Comprehension by Computer: Expectation-Based Analysis of Sentences in Context. Tech. Rept. 78, Computer Science Dept., Yale University, 1976.

56. Ross, J. R. "Metaanaphora." *Linguistic Inquiry* (1970).

57. Sacerdoti, E. D. Language Access to Distributed Data with Error Recovery. Proc. Fifth Int. Jt. Conf. on Artificial Intelligence, MIT, 1977, pp. 196-202.

58. Schank, R. C. and Colby, K. M. (Eds.). *Computer Models of Thought and Language.* Freeman, San Francisco, 1973.

59. Schank, R. C.. *Conceptual Information Processing.* Amsterdam: North-Holland, 1975.

60. Schank, R. C. and Abelson, R. P.. *Scripts, Goals, Plans and Understanding.* Hillside, NJ: Lawrence Erlbaum, 1977.

61. Schank, R. C. and Carbonell, J. G. Re The Gettysburgh Address: Representing Social and Political Acts. In *Associative Networks*, Findler, Ed.,New York: Academic Press, 1979.

62. Schank, R. and Riesbeck, C.. *Inside Computer Understanding.* Lawrence Erlbaum Associates, New Jersey, 1980.

63. Schank, R. C., Lebowitz, M., Birnbaum, L. "An Integrated Understander." *American Journal of Computational Linguistics 6*, 1 (1980), 13-30.

64. Searle, J. R.. *Speech Acts.* Cambridge University Press, 1969.

65. Searle, J. R. Indirect Speech Acts. In *Syntax and Semantics, Volume 3: Speech Acts*, P. Cole and J. L. Morgan, Eds., New York: Academic Press, 1975.

66. Sidner, C. L. Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse. Tech. Rept. TR-537, MIT AI Lab, Cambridge, Mass., 1979.

67. Simmons, R. F. Semantic Networks: Their Computation and Use for Understanding English Sentences. In *Computer Models of Thought and Language*, Schank, R. C., and Colby, K. M., Ed.,Freeman, San Francisco, 1973, pp. 63-113.

68. Small, S. L. and Rieger, C. Parsing and Comprehending with Word Experts (A Theory and its Realization). In *Strategies for Natural Language Processing*, Ringle and Lehnert, Ed.,Lawrence Erlbaum, 1982.

69. Small, S., Cotrell, G., and Shastri, L. Toward Connectionist Parsing. AAAI-82, Univ. of Pittsburgh, Pittsburgh, August, 1982, pp. 247-250.

70. Waltz, D. L. and Goodman, A. B. Writing a Natural Language Data Base System. IJCAIVproc, IJCAI-77, 1977, pp. 144-150.

71. Weischedel, R. M. and Black, J. "Responding to Potentially Unparseable Sentences." *American Journal of Computational Linguistics 6* (1980), 97-109.

72. Weizenbaum, J. "ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine." *Comm. ACM 9*, 1 (Jan. 1966), 36-45.

73. Wilks, Y. A. Preference Semantics. In *Formal Semantics of Natural Language*, Keenan, Ed.,Cambridge University Press, 1975.

74. Winograd, T.. *Language as a Cognitive Process, Volume I: Syntax*. Addison Wesley, 1982.

75. Woods, W. A. "Transition Network Grammars for Natural Language Analysis." *Comm. ACM 13*, 10 (Oct. 1970), 591-606.

76. Woods, W. A., Kaplan, R. M., and Nash-Webber, B. The Lunar Sciences Language System: Final Report. Tech. Rept. 2378, Bolt, Beranek, and Newman, Inc.. Cambridge, Mass., 1972.

77. Woods, W. A., Bates, M., Brown, G., Bruce, B., Cook, C., Klovstad, J., Makhoul, J., Nash-Webber, B., Schwartz, R., Wolf, J., and Zue, V. Speech Understanding Systems - Final Technical Report. Tech. Rept. 3438, Bolt, Beranek, and Newman, Inc., Cambridge, Mass., 1976.